Anomaly Detection in Digital Substation Communication Using Transformer-Based Distribution Fitting

Himanshu Goyel, *Graduate Student Member, IEEE*, Alfan Presekal, Peter Palensky, *Senior Member, IEEE*, and Alexandru Ştefanov, *Member, IEEE*

Abstract*—Digital substations, which replace traditional analog infrastructure, are essential to power grid operation but are facing growing vulnerability to cyber attacks. Existing anomaly detection in substation communication requires labeled datasets for supervised training and fails to incorporate temporal characteristics, which cannot detect unknown persistent attacks. Setting arbitrary thresholds for outlier detection leads to high false positives and low detection rates. This paper addresses cyber security challenges related to IEC 61850 Generic Object Oriented Substation Event (GOOSE) protocol within digital substations. We propose a novel unsupervised Transformer-based Distribution Fitting Anomaly Detection (TF-DiFAD) method for time series GOOSE frames with a robust thresholding technique. Deep packet inspection is used to extract features from GOOSE frames, which are processed through the proposed TF-DiFAD model. TF-DiFAD combines the deep learning transformer model with statistical distribution fitting techniques to accurately detect anomalous GOOSE frames. Specifically, reconstruction errors are generated using a state-of-the-art transformer model. A novel model-agnostic solution is applied for setting anomaly thresholds and calculating anomaly probabilities. The Kolmogorov-Smirnov test is employed to select the best-fitting distribution for these errors. TF-DiFAD is benchmarked against other state-of-the-art models using two distinct test datasets, demonstrating superior performance. The results indicate that TF-DiFAD detects anomalies with Receiver Operating Characteristics Area Under Curve (ROC AUC) scores of 96.84% and 95.73% respectively for both datasets.

***Index Terms*—Anomaly detection, deep learning, digital substation, distribution fitting, statistics, transformer.**

I. Introduction

DIGITAL substations are critical components of modern power grids, replacing conventional analog systems with advanced digital infrastructures. This enables significant improvements in operational efficiency, as well as real-time monitoring, automation, protection, and control capabilities [1]. However, the increased power grid digitalization has introduced considerable cyber security vulnerabilities with digital substations being exposed to a broader range of cyber threats [2]. Cyber attacks on power grids could often target the integrity and availability of critical monitoring and control functions. This includes targeting Wide Area Monitoring Systems (WAMS) [3] and state estimation algorithms [4] via phasor measurement units, DC microgrids via power converter based deception and replay attacks [5], power grid connected PV systems via false data injection attacks [6] or digital substations via Generic Object Oriented Substation Event (GOOSE) packet manipulation.

The motivation for this research stems from the critical gap in existing studies on anomaly detection in substation communication networks. Most existing works depend on supervised approaches [7], [8], [9] that necessitate labeled datasets. Moreover, they fail to incorporate the temporal characteristics of the messages, or focus on communication network throughput and high-level Operational Technology (OT) network features rather than analyzing entire messages through payload parsing. While observing communication network flow-level characteristics alone may be sufficient for detecting OT network flow intensive attacks, e.g., Distributed Denial of Service (DDoS), these methods fail in detecting sophisticated, protocol-specific injection attacks which do not cause OT network flow changes [10]. Furthermore, our analysis indicates that existing anomaly detection methods assume the reconstruction error is normally distributed. They lack a generalized approach for setting detection thresholds and calculating anomaly probabilities. Consequently, the existing literature either sets the thresholds arbitrarily or uses three-sigma rule based on normal distribution. This limits the model performance, as the reconstructed error does not necessarily follow a normal distribution.

This research addresses the identified gaps directly. We propose a novel method for temporal anomaly detection in digital substations using GOOSE deep packet inspection with an unsupervised transformer-based model for reconstruction and distribution fitting for threshold setting.

The proposed model is based on transformer Deep Learning (DL) technique and statistical distribution fitting to detect anomalies in IEC 61850 GOOSE data frames for digital substations. It distinguishes anomalies from normal network behavior in IEC 61850 GOOSE communications. This approach aligns with the cyber security guidelines outlined in the IEC 62351 standard [11], which is specifically developed to secure communication protocols in power systems.

The proposed anomaly detection method explicitly considers the temporal aspect of the GOOSE data frames and trains

Received 10 June 2025; revised 24 November 2025; accepted 6 January 2026. Date of publication 12 January 2026; date of current version 23 June 2026. This work was supported by the EU Horizon Europe Cooperative Cyber Protection for Modern Power Grids (COCOON) Project under Grant 101120221. Paper no. TSG-01145-2025. *(Corresponding author: Alexandru Ştefanov.)*

The authors are with the Department of Electrical Sustainable Energy, Delft University of Technology, 2628 CD Delft, The Netherlands (e-mail: H.Goyel@tudelft.nl; A.Presekal@tudelft.nl; P.Palensky@tudelft.nl; A.I.Stefanov@tudelft.nl).

Digital Object Identifier 10.1109/TSG.2026.3652798

in an unsupervised fashion, making it applicable to both real-time monitoring and offline analysis. The proposed technique also provides a generalized model-agnostic threshold setting mechanism for individual features. This research contributes to strengthening the cyber security of digital substations and supports the broader objective of protecting digitalized power grids. The key scientific contributions of this paper are summarized as follows:

1) We propose a novel unsupervised anomaly detection method using GOOSE deep packet inspection and a transformer-based algorithm that accounts for temporal aspects of the data frames.
2) We develop a novel, comprehensive, and model-agnostic technique to calculate thresholds for anomaly prediction and corresponding anomaly probabilities. By fitting the reconstruction error from the DL model to distribution probabilities, we overcome the prevalent hypothesis that these errors follow a normal distribution.
3) We perform extensive empirical experiments on two publicly available GOOSE datasets using a two-phase training of the reconstruction model, i.e., pre-training and fine-tuning, to reduce the effects of random initialization bias.
4) We benchmark the proposed TF-DiFAD against state-of-the-art methods including Convolutional Neural Network [12] (CNN), Ensemble learning [13], Long-Short Term Memory [14] (LSTM) and ResNeST [15]. The comparative analysis across multiple performance metrics demonstrates TF-DiFAD consistently outperforms these models, achieving ROC AUC scores of 96.84% and 95.73%, respectively for both datasets.

We limit the scope of the work to anomaly detection in GOOSE as opposed to prevention. The model is designed to be connected in parallel to the OT network similar to an Intrusion Detection System (IDS). This contrasts with an intrusion prevention system, which connects in line to actively block malicious messages. The primary goal of the architecture is to inform the system operator of malicious events in substation OT networks to initiate response strategies. The model alerts the system operator of anomalous device details within a few seconds, i.e., 1-5 secs. The proposed model can be deployed in the substation OT network on a server in a containerized environment or a dedicated device capable of monitoring live network traffic.

The paper is structured as follows. Section II discusses the related works and identified research gaps. Section III describes the proposed TF-DiFAD model, including feature selection, reconstruction error generation, probability distribution fitting, and reconstruction model training. Section IV presents the anomaly detection performance evaluation, and Section V discusses the conclusions and future work.

II. Related Works and Research Gaps

IDS could be deployed in digital substations to monitor OT communication networks and identify cyber attacks. The primary objective of anomaly detection-based IDS is to differentiate between normal and abnormal behavior in OT communication network traffic, by recognizing deviations from established patterns. Anomaly detection enhances the overall cyber security and integrity of OT communications in digital substations [16].

TABLE I
Related Work Comparison

Work (Citation)	Model type	Key Limitation(s)
D. Jay et al. [20]	DBSCAN & Autoencoder	Arbitrary threshold; no temporal context.
A. Nassar et al. [7]	CWT + CNN	Supervised (labelled)
V. Quincozes et al. [8]	GOOSE feature engineering.	Supervised (labelled)
X. Wang et al. [9]	BiLSTM	Supervised; (labelled)
I. Ji et al. [21]	Autoencoder + LSTM	No temporal context
W. Hao et al. [22]	FARIMA (forecast-based) for DDoS.	Ineffective for GOOSE
T. Yang et al. [15]	ResNeST for DDoS.	Ineffective for GOOSE
Y. M. Khaw et al. [12]	Unsupervised CNN Autoencoder	Arbitrary threshold; ignores network-level data.
A. Albarakati et al. [13]	Unsupervised ensemble	No robust threshold
A. Presekal et. al [23]	EGC-LSTM.	Semi-supervised; not substation-specific.
Current work	Transformer + Distributed fitting	Unsupervised, temporal context, substation and GOOSE specific, considered payload, robust threshold

Anomaly detection at a high level can be split into model-driven and data-driven detection. While model driven anomaly detectors [5] are fast and effective against known attacks, they are less effective against zero-day attacks. However, data-driven anomaly detection methods are more efficient against known and zero-day attacks. Within data-driven anomaly detection, Darban et al. [17] classify anomaly detection into three major types which are: 1) forecasting-based, where the model forecasts the inputs for the next time stamp; 2) reconstruction-based, where the model reconstructs the input for the current time stamp; and 3) representation-based, where the model is trained and the latent space rather than output is used for anomaly detection.

Machine Learning (ML) and DL techniques for anomaly detection have gained significant attention over the past decade due to their superior performance. Supervised DL methods require labeled datasets to distinguish between normal and anomalous instances, enabling superior performance. However, a major limitation of these approaches is the dependency on labeled datasets [18]. In contrast, unsupervised DL methods do not require labeled data. Instead, they detect anomalies by identifying deviations from normal behavior. This makes them suitable for real-world scenarios where anomalous data is infrequent, evolving, or difficult to define in advance. Unsupervised methods focus on learning what constitutes normal behavior, which is generally more feasible than anticipating and labeling every possible anomalous scenario [19].

State-of-the-art research has proposed methods for anomaly detection in digital substations. Table I outlines the research gaps in the current literature and highlights the major contributions of this paper. We discuss these works and their limitations in this section.

Jay et al. [20] propose an attack detection method for GOOSE frames using DBSCAN and autoencoders. The proposed model employs a three-agent approach to identify malicious activity in the data link layer, Application Protocol

Data Unit (APDU), and payload. While effective against the dataset considered in the paper, the model was not evaluated against any other datasets. Additionally, the anomaly detection threshold was set arbitrarily, lacking a systematic approach. Furthermore, the model uses individual data points as input rather than windowed data, thus neglecting the temporal context in its analysis.

Abu Nassar et al. [7] present a technique for detecting cyber attacks and power disturbances in substations. Their method utilizes Continuous Wavelet Transforms (CWT) to extract three key features from a total of 29 available features, which are then analyzed using a CNN to identify cyber attacks. While the methodology has been tested on three datasets from simulated substation environments and validated in real-time using OPAL-RT, it relies on a supervised learning approach for attack detection. As mentioned previously, the supervised approach needs labelled datasets for both normal and abnormal data which is impractical in real world scenarios. Quincozes et al. [8] discuss the feature engineering in GOOSE traffic identifying features that might be useful for detecting intrusions in a substation. But similar to [7] the proposed method also uses a supervised approach of attack detection.

Wang et al. [9] propose a BiDirectional Long Short-Term Memory (BiLSTM) network combined with a sliding window approach for anomaly detection in substation network traffic using GOOSE datasets. While the model effectively leverages labeled datasets to identify anomalies, this reliance on labeled data limits its applicability in real-world where labeled attack data may be scarce. Furthermore, the study primarily concentrates only on detecting false data injection attacks.

Ji et al. [21] propose an unsupervised approach for detecting attacks in a DNP3-based digital substation network, utilizing a combination of autoencoder and LSTM networks. Their model leverages deep packet inspection in DNP3 to identify anomalies within the OT network traffic. While the LSTM effectively captures long-term dependencies, a notable limitation of the approach is its failure to incorporate temporal sequences for learning periodic short-term dependencies using a sliding window approach. Instead, the model processes single-point inputs rather than using time series data.

Hao et al. [22] propose the use of Fractional Autoregressive Integration Moving Average (FARIMA) to detect abnormal patterns in substation communication networks. The model considers only network traffic data, such as data transfer rate and packet inter-arrival time, to detect multiple DDoS attacks. The model uses a forecast-based approach rather than a reconstruction-based approach. While this model is suitable for detecting DDoS-based attacks where the OT network traffic is increased or decreased significantly, it is ineffective against GOOSE specific attacks where a crafted packet is injected into the OT network. Yang et al. [15] propose a feature extraction and ResNeST model to detect DDoS attacks in substations. The model uses only high-level frequency domain network traffic data to detect DDoS attacks, which is not suitable for detecting GOOSE-specific attacks.

Khaw et al. [12] propose an unsupervised CNN-based autoencoder to detect tripping attacks in IEC 61850-based digital substations, considering multiple attack scenarios that can trigger relay tripping. However, the paper sets a threshold of 1.5 times the maximum reconstruction error without any justification for this. Additionally, it only considers payload inputs directly from Current transformers (CTs) and Voltage Transformers (VTs), excluding network-level data. This lack of network-level data, along with payload information, may limit the features available for attack detection and prevent detection of GOOSE spoofing attacks, such as suppression attacks.

Albarakati et al. [13] employ an unsupervised approach by deploying a substation model and detecting cyber attacks using an ensemble, prediction-based algorithm for GOOSE-based attacks. The model combines LSTM, Gated Recurrent Units (GRU), and Recurrent Neural Network (RNN) in an encoder-decoder structure with a bagging algorithm to detect attacks in the digital substation. Although it uses a moving window to predict the sequence's next value, the threshold definition is not robust. The paper proposes a threshold of an arbitrary constant α times the maximum validation loss, but α is not defined. This unstructured selection of α by the operator can be difficult, as a high α may cause missed attacks.

In one of our previous research [14], we propose a Graph Convolution-based LSTM (GC-LSTM) combined with a CNN for detecting anomalous nodes in OT communication networks. LSTM is used to predict the original OT network traffic and train a convolutional network to detect cyber attacks. The model uses OT network traffic throughput as input to the GC-LSTM model and doesn't use the packet or payload data. In an extended study [23], we introduce an Enhanced Graph Convolution LSTM (EGC-LSTM), which is a semi-supervised approach for cyber threat detection. Unlike the earlier model, this approach considers both the payload and network traffic characteristics of OT network data. However, it is still dependent significantly on labelled datasets. Additionally, the model doesn't consider digital substation specific packets; rather, it focuses on OT traffic from wide area networks.

Thus, the current literature as outlined in Table I reveals a clear gap in research regarding unsupervised anomaly detection techniques that leverage GOOSE payload and frame information, combined with using a robust and comprehensive threshold setting algorithm, for identifying anomalies in digital substations. To address this gap, we propose the TF-DiFAD model to overcome the limitations faced by existing anomaly detection methods. Furthermore, we evaluate the proposed model's performance against several of these algorithms, which are modified appropriately to suit the dataset and experimental setup.

III. Anomaly Detection Using Transformer-Based Distribution Fitting

The proposed TF-DiFAD model employs a one-class unsupervised anomaly detection method to detect cyber attacks exploiting vulnerabilities of the IEC 61850 GOOSE protocol. The detection algorithm is trained only on normal GOOSE frames and does not utilize anomaly data for training. Consequently, the model defines detection boundaries solely on normal GOOSE behavior.

The model is divided into three stages, which are: 1) Feature Selection, where we identify relevant features from the GOOSE frames, 2) Reconstruction Error Generation, where we generate the reconstructed feature representations of the

GOOSE frame; and 3) Probability Distribution Fitting, where we identify the probability distribution of the reconstruction errors to enable the definition of boundaries and classify GOOSE data frames as normal or anomalous.

A. Feature Selection

A GOOSE frame consists of several components, which are explained in detail in [24]. Along with the GOOSE frame *timestamp*, our model uses deep packet inspection to extract the following information from the GOOSE APDU:

1) *goID*: GOOSE identifier specifies a unique identification for the GOOSE control block.
2) *stnum*: State number increments in case a change in one of the GOOSE dataset members is detected.
3) *sqnum*: Sequence number increments with every new frame and resets to 0 with an increment in *stnum*.
4) *allData*: Transmitted dataset that contains the physical information.

According to the IEC 61850 standard for GOOSE protocol, in case of an event, the rate of GOOSE stream increases while the status number increments, and the sequence number resets to 0. The allData value contains the change itself, either in the form of a Boolean changing from true to false, which could represent the position of a circuit breaker, or a float value change, which could represent the measurement.

After feature selection, the entire dataset is segmented based on the unique *goID*s since each GOOSE stream is identified by its own distinct *goID*. For each unique *goID* dataset, the following features are computed as illustrated in Fig. 1.

1) **Packet Arrival Rate**: Calculated as $(t_i^{arr} - t_{i-1}^{arr})^{-1}$ where, t_i^{arr} is the current frame's arrival time and t_{i-1}^{arr} is the previous packet's arrival time.
2) **Change in stnum**: Calculated as $stnum_i - stnum_{i-1}$ where, $stnum_i$ is the current frame's state number and $stnum_{i-1}$ is the previous frame's state number.
3) **Change in sqnum**: Calculated as $sqnum_i - sqnum_{i-1}$ where, $sqnum_i$ is the current frame's sequence number and $sqnum_{i-1}$ is the previous frame's sequence number.
4) **Type flag**: Indicates the type of data in *allData*. 0 represents only Boolean, 1 represents only float, 2 represents both Boolean and float, 3 represents any other possible combination.
5) **Boolean values**: Shows the actual Boolean values in the *allData* part of the frame.
6) **Float values**: Show the actual float values in *allData*.

We calculate the changes in the values to maintain better numerical stability during the model training. This is because *stnum* and *sqnum* are continuously increasing and could result in arbitrarily large numbers that were never part of the original training dataset.

These features are normalized using standard normal scaler, and they form the multivariate inputs $X_{T,k}^n$ for the next part of the model, where $X_{T,k}^n$ is a three-dimensional vector with T being the total number of data points in *goID* n with a dimension of k.

B. Reconstruction Error Generation

The overall architecture of the proposed model after feature extraction can be seen in Fig 2. From this point onward,

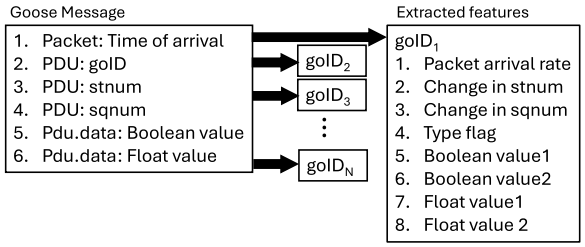


Fig. 1. Feature selection for each GOOSE frame.

we consider data processing for a single *goID*, and a similar process is repeated for each *goID*. Essentially, $X_{(T,k)}$ which is a 2-dimensional vector is considered instead of $X_{T,k}^n$ which is 3-dimensional vector by dropping *goID* reference for simplicity.

Considering a multivariate time series vector $\{x_1, x_2, \ldots, x_T\}(X \in \mathbb{R}^{T\times k})$ where $x \in \mathbb{R}^k$ is a k dimensional feature vector, we build a reconstruction model which outputs the reconstruction error $e = \{e_t, e_{t+1}, \ldots, e_T\}$ where $e \in \mathbb{R}^{(T-t+1)\times k}$ considering a context length of t. The reconstruction error e captures the latent patterns within X and has a higher reconstruction error if the input deviates from its latent patterns.

Given an input ($X \in \mathbb{R}^{T\times k}$), we define a window length t and stride of s. In this work we are considering a simple case of stride s= 1, although we could use larger strides that may be used in practice to reduce computation burden by spacing windows further apart. We extract the windows (w_i) of length t by sliding a window of size t over X. The i-th window w_i can be given as:

$$w_i = \{x_i, x_{i+1}, \ldots, x_{i+t-1}\} \text{ for } i = 1, \ldots, \quad T - t + 1$$

where, $x_i \in \mathbb{R}^k$ implies $w_i \in \mathbb{R}^{t\times k}$. Collecting all the windows into a single array W gives:

$$W = \{w_1, w_2, \ldots, w_{T-t+1}\} \Rightarrow W \in \mathbb{R}^{b\times t\times k}$$

where b is the total number of windows given by $b = T - t + 1$. The sliding window is used to set the context length for the model, allowing it to learn the temporal dependencies. The output of the sliding window W is then passed through a SoftMax layer, which applies the SoftMax operation over the dimension t and returns a vector $I \in \mathbb{R}^{b\times t\times k}$ as in (1).

$$I_{l,i,j} = \frac{\exp\left(W_{l,i,j}\right)}{\sum_{\alpha=1}^{t} exp\left(W_{l,\alpha,j}\right)}, \quad \forall l \in \{1, 2, \ldots, b\}\, i \in \{1, 2, \ldots, t\}\, j \in \{1, 2, \ldots, k\} \tag{1}$$

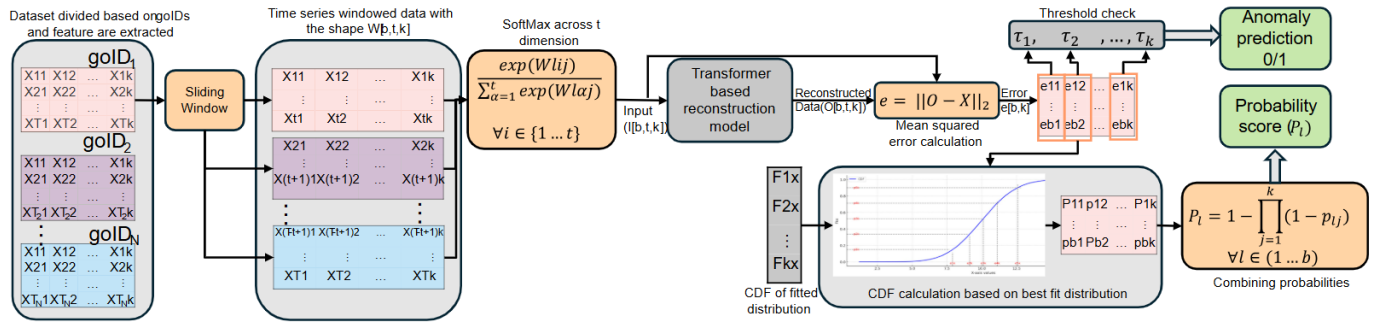
Lemma 1: Applying a SoftMax layer to a sequence removes the constant offset within the window, allowing focus on only the relative changes in feature values.

Proof: Let a signal $X = \{x_1, x_2, \ldots, x_t\}$ represents data from a single window. Assume each data point in the signal can be expressed as a sum of two parts, a constant part C and a variable part v. Thus, the signal can be represented as,

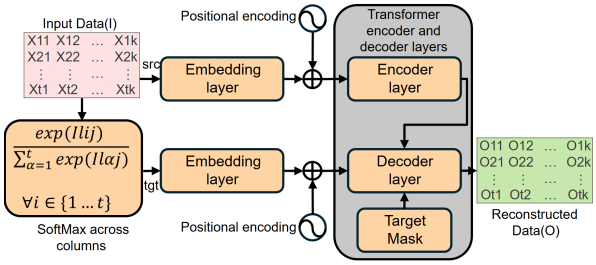
$$X = \{x_1, x_2, \ldots, x_t\}$$
$$X = \{C + v_1, C + v_2, \ldots, C + v_t\}$$

Calculating the SoftMax of the signal gives us,

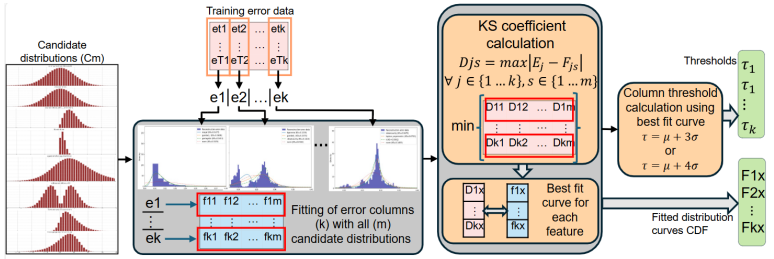
$$I_i = \frac{exp(X_i)}{\sum_{\alpha=1}^{t} exp(X_\alpha)}, \quad i \in \{1, 2, \quad \ldots, t\}$$



(a)TF-DiFAD model: Dataset divided based on *goID*, applied sliding-window to form time-series segments which are then passed through a column-SoftMax layer, transformer-based reconstruction module, and a distribution-fitted CDF to get probability score and prediction.



(b)Transformer architecture used with full encoder and decoder.



(c) Fitting reconstruction error of training data on candidate distributions and K-S test.

Fig. 2. TF-DiFAD architecture with sliding window, SoftMax, transformer, reconstruction and distribution fitting.

$$= \frac{exp(C + v_i)}{\sum_{\alpha=1}^{t} exp(C + v_\alpha)} = \frac{exp(C).exp(v_i)}{\sum_{\alpha=1}^{t} exp(C).exp(v_\alpha)}$$

$$I_i = \frac{exp(v_i)}{\sum_{\alpha=1}^{t} exp(v_\alpha)}, \quad i \in \{1, 2, \quad \ldots, t\}$$

Thus, Lemma 1 shows that the SoftMax layer removes any constant offset in the window, enhancing numerical stability.

The SoftMax function, exaggerates the largest values in the incoming window. In an attack case where high-magnitude packets are injected into a window of normal, low-magnitude packets, the SoftMax will assign values closer to 1 to the attack packets while values closer to 0 for the normal packets. This creates a pattern that is significantly deviating from the normal patterns. While it is true that an attack filling the entire window with a constant high magnitude might appear uniform after SoftMax, such an attack is not a blind spot. Detection will occur at the "rising edge" the moment the first anomalous packet creates this high magnitude and again at the "falling edge" when normal packets are entering back in the window. Thus adding a SoftMax layer ensures that the transformer subsequently used learns from the relative changes in the data points rather than absolute values while also remaining numerically stable. Thus justifying the use of SoftMax layer in Fig 2a.

In the next part of the method, although any sequence-to-sequence model can be employed for reconstruction, we use a transformer model because of its better attention mechanism and state-of-the-art performance. The output from the SoftMax layer is passed to the transformer, which follows the structure shown in Fig. 2b. Following Vaswani et al. [25], both encoder and decoder comprise multi-head attention and feed-forward layers, and an attention mask which is applied to the decoder. The transformer model accepts two inputs, referred to as *src* and *tgt*. Here, *src* is the SoftMax output $I \in \mathbb{R}^{b\times t\times k}$ while *tgt* is I again passed through a SoftMax layer. The output of the model is the reconstructed input data by the model and is given by $O \in \mathbb{R}^{b\times t\times k}$.

Finally, we perform a Mean Squared Error (MSE) of the output $O \in \mathbb{R}^{b\times t\times k}$ from the transformer reconstruction model and the input to the model $I \in \mathbb{R}^{b\times t\times k}$ across the dimension t to get the reconstruction error $e \in \mathbb{R}^{b\times k}$ as shown in (2).

$$e_{l,j} = \sqrt{\sum_{i=1}^{t} \left(O_{l,i,j} - I_{l,i,j}\right)^2}, \quad \forall l \in \{1, 2, \ldots, b\}\ j \in \{1, 2, \ldots, k\} \tag{2}$$

Thus, we finally have a reconstruction error $e \in \mathbb{R}^{b\times k}$ generated from inputs $I \in \mathbb{R}^{T\times k}$ where, $b = T - t + 1$ if no padding is applied to the input data.

C. Probability Distribution Fitting

This subsection presents how to translate the reconstructed errors obtained from the previous model to anomalies and probabilities. The proposed probability distribution fitting method uses reconstruction errors from the transformer DL model as input. However, this is not a constraint. The reconstruction error can originate from any model, making the proposed method model agnostic.

We divide this process into two parts which are Detection and Training. We explain both the detection and training modules in detail.

1) Detection: As seen in Fig. 2a, the error e passes through the detection phase, where the errors e are compared with the predefined thresholds $\tau = \{\tau_1, \tau_2, \ldots, \tau_k\}, \tau \in \mathbb{R}^k$ to check if the error is an anomaly or not. This τ is obtained from the training phase explained in the subsequent subsection. In case

of any violations in thresholds, we mark the whole row as an anomaly, as in (3).

$$A_i = \begin{cases} 1, & \text{if} \quad \exists \quad j : e_{lj} > \tau_j \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Further, given predefined Cumulative Distribution Functions (CDFs) obtained from the training phase:

$$F = \{F_{1x}, F_{2x}, \ldots, F_{kx}\}, F \in \mathbb{R}^k,$$

where each F_{jx} corresponds to the best-fitting distribution selected from m candidate distributions for the j-th dimension. The identification of F_{jx} is explained in the subsequent training subsection. Now to calculate the probability of a sequence being an anomaly, we consider an event $Q_{l,j} = \{X_j < e_{lj}\}$, which represents the scenario where a random variable X_j is less than the observed error e_{lj}. The probability of each event Q_{lj} is computed as

$$p_{l,j} = P(Q_{lj}) = F_{jx}(e_{lj}), \quad \forall l \in \{1, 2, \ldots, b\}\, j \in \{1, 2, \ldots, k\}$$

Considering e_{lj} follows the corresponding distribution F_{jx}. Thus, we now have the probabilities p_{lj} corresponding to each error component e_{lj}.

We make the assumption that errors $(e_{l1}, e_{l2}, \ldots, e_{lk})$ are independent variables similar to the one taken in Naïve Bayes classification technique. This is a simplification, as modeling the joint probability distribution of all the errors could add significant complexity while simultaneously reducing the combining interpretability further. Thus considering this assumption we combine these probabilities to obtain the overall probability P_l that at least one of the events Q_{lj} occurs:

$$P_l = P\left(\bigcup_{j=1}^{k} Q_{lj}\right) = 1 - \prod_{j=1}^{k}(1 - p_{l,j}), \quad \forall l \in \{1, 2, \ldots, b\}$$

Finally, P_l represents the probability that at least one of the k random variables $(X_1, X_2, \ldots, X_k)$ is less than the corresponding observed errors $(e_{l1}, e_{l2}, \ldots, e_{lk})$. The combined probability P_l is used as the anomaly score. The anomaly score P_l for each time window l is calculated as in (4).

$$P_l = 1 - \prod_{j=1}^{k}[1 - F_{j,x}(e_{l,j})], \quad \forall l \in \{1, 2, \ldots, b\} \tag{4}$$

It is important to note that the calculated probability P_l from (4) is used as a score for classification, not as a perfectly calibrated probability. While the true overall anomaly probability $P_l = P\left(\bigcup_{j=1}^{k} Q_{lj}\right)$ could be calculated considering the correlation between the features, this is currently out of the scope of the paper.

2) Training: Figure 2c shows the process used to train or generate the thresholds $\tau \in \mathbb{R}^k$ and predefined CDFs $F \in \mathbb{R}^k$. In this subsection we show how we calculate the thresholds and best fitted CDFs. We consider a set of m different parametric candidate distributions, each with a distinct Probability Density Function (PDF), denoted by C_s where $s \in 1, \ldots, m$. We use a training set of error vectors e_{lj}, obtained from the reconstruction error module. Let C_s be the PDF of s-th candidate distribution, parameterized by θ. We estimate the parameters $\hat{\theta} \in \mathbb{R}^{k \times m}$ for k error vectors and m candidate distributions using the Maximum Likelihood Estimator (MLE) as below:

$$\hat{\theta}_{j,s} = \text{argmax}_\theta \sum_{l=1}^{b} \log(C_s(e_{lj};\theta)) \quad \forall j \in \{1, \ldots, k\}\, s \in \{1, \ldots, m\}$$

Once we obtain the MLE parameters, we define the fitted PDF function $f_{j,s}(x)$ as below

$$f_{j,s}(x) = C_s(x; \hat{\theta}_{j,s}), \quad \forall j \in \{1, \ldots, k\}, s \in \{1, \ldots, m\}$$

where $C_s(x;\hat{\theta}_{j,s}))$ is the probability density of candidate distribution with parameters $\hat{\theta}$ estimated by MLE. Let $F_{j,s}(x)$ be the CDF for corresponding PDFs $f_{j,s}(x)$ given by,

$$F_{j,s}(x) = \int_{-}^{} \infty^x f_{j,s}(u)du, \quad \forall j \in \{1, \ldots, k\}, s \in \{1, \ldots, m\}$$

Moving forward, we perform the Kolmogorov-Smirnov (K-S) test [26] and calculate the K-S coefficient, $D_{j,s}$ for all the fitted distributions. The K-S test is a non-parametric test which is used to measure dissimilarity between two CDFs. The K-S statistics are given by,

$$D_{j,s} = \max_x |E_j(x) - F_{j,s}(x)|$$

where $E_j(x)$ is the empirical cumulative distribution function (ECDF) computed from error vector $e_j \in \mathbb{R}^b$ corresponding to the j-th dimension for $j \in \{1, \ldots, k\}$.

Now we have $D_{j,s}$ values organized in a matrix with k rows, i.e., features, and m columns, i.e., candidate distributions. We select the minimum across all m distributions to get the best K-S statistics for each feature, giving us $D_{j,x}$ where $j \in 1, \ldots, k$ and x denotes the index of the distribution with best (smallest) K-S for that feature. Finally, the corresponding CDF is selected as the best-fitted CDF (F_{j,x_j}). This can be given in the form as below:

$$x_j = \text{argmin}_s D_{j,s}, \text{corresponding} F_{j,x_j} \quad \text{is selected}$$

We define F_{j,x_j} as $F_{j,x}$ for simplicity and $f_{j,x}$ as corresponding PDF. Although the algorithm is implemented in a step-by-step manner as described above, for completeness, the final expression for the best fitted CDF $F_{j,x}$ can be written in terms of input reconstruction errors $e_{l,j}$ and candidate PDF C_s as in (5).

$$F_{j,x}(x) = F_{j,x_j}(x) = \int_{-\infty}^{x} C_{x_j}\left(u;\hat{\theta}_{j,x_j}\right) du$$

where:

$$x_j = \text{argmin}_s \max_x |E_j(x) - \int_{-\infty}^{x} C_s(u;\hat{\theta}_{j,s})du$$

$$\hat{\theta}_{j,s} = \text{argmax}_\theta \sum_{l=1}^{b} \log\left(C_s\left(e_{l,j};\theta\right)\right)$$

$$E_j = \frac{1}{b}\sum_{l=1}^{b} 1_{e_{l,j} \le x} \quad j \in 1, \ldots, k \tag{5}$$

So we have the best fitted CDFs for all dimension k, further we see how we define the thresholds τ, we start with the conventional Gaussian-based anomaly definition, where the threshold $(\tau_j, j \in 1, \ldots, k)$ is defined as,

$$\tau_j = \mu_j \pm 3\sigma_j$$

Here, μ_j is the expected value of the variable and σ_j is the standard deviation (root of variance). In a Normal distribution, this region would encompass 99.73% of the data, and anything outside of this would qualify as an anomaly. Also, for a Normal symmetric distribution, μ would be the mean and the median. However, since we are moving away from Normal and symmetric distributions, we intend to find a threshold that captures the same probability mass as $\mu + 3\sigma$ using the fitted distribution's CDF. This is done to have a generalized realistic default threshold which can be used to identify outliers in non-Gaussian distributions. In case of non-Gaussian distributions we still follow the norm considering 99.865^{th} percentile as normal data and above that as anomaly (approx. 0.135%). This makes sure that the model is comparable to the existing practice of $\mu+3\sigma$. However, the percentile or probability mass could be adjusted based on operational risks.

Moving ahead, let τ_j be the upper threshold which encompasses the $\mu + 3\sigma$ aspect of the distribution, which can be given for a non-Normal distribution as,

$$F_{j,x}(\tau_j) = \int_{-\infty}^{\tau_j} f_{j,x}(x)dx = 0.5 + \frac{R}{2}$$

Here R is the region considered for majority of data points outside of which the data will be considered anomaly. In the case of $\mu + 3\sigma$, R would be 0.998650.

Assumption 1: Considering a model is accurately able to reconstruct the input, the reconstruction error would be zero. This would imply there is no need for a lower limit threshold.

Based on Assumption 1, we do not define or enforce a lower threshold. Finally, we can find the final thresholds $(\tau_j, j \in \{1, \ldots, k)$ using (6) which generalizes the concept of "upper threshold" for anomaly detection to Non-Gaussian distributions using the inverse CDF of the best fitted distribution.

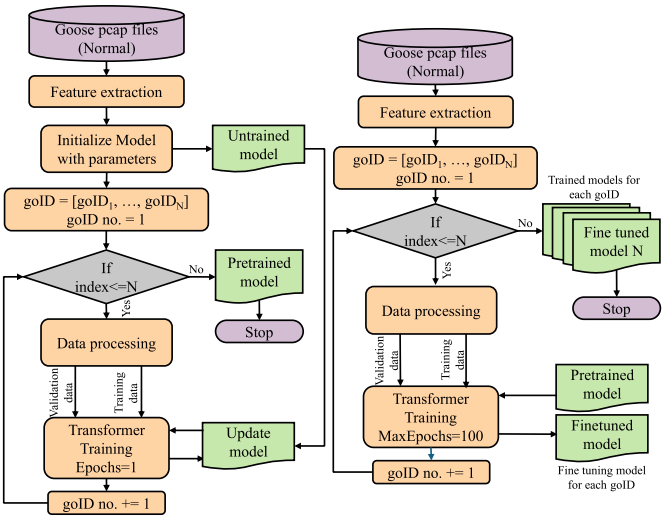
$$\tau_j = F_{j,x}^{-1}\left(\frac{R+1}{2}\right) \tag{6}$$

We calculate the thresholds τ_j considering probability mass of $\mu + 3\sigma$ and $\mu + 4\sigma$.

D. Two-Phase Training and Detection

As stated earlier, we train a unique Transformer model to reconstruct the input for each *goID*. The training process is divided into two phases: pre-training and fine-tuning. In the pre-training phase, a randomly initialized model is trained on a large general dataset, and fine-tuning is when the pre-trained model is further trained for a specific dataset for capturing unique characteristics of the dataset.

Pre-training + Fine-tuning is considered to make the model more generalized and help prevent the model from getting stuck in a local minima with high probability [27]. Pretraining also helps the model to converge faster [28] and prevents it from diverging, which could be the case for a randomly initialized model. Thus, we chose this structure of *Pre-training+Fine-tuning* for training our model.



(a) Phase 1: Pre-training (b) Phase 2: Fine-Tuning

Fig. 3. Two-step training of the transformer model using initial pre-training over all *goID*s and later fine tuning model over individual *goID*s.

1) Phase 1: Pre-Training: We implement the two phases as shown in Fig. 3. In the pre-training phase as shown in Fig. 3a, we select approximately 70% of normal GOOSE pcap files which are passed through the feature extraction module. The model is initialized with random parameters to produce an untrained baseline model. Further, the datasets are grouped based on *goID*s, and an iterative loop is executed for each *goID*. The data is processed by dividing it into a validation and a training dataset. It is normalized, which is then passed through the sliding window and SoftMax. The model parameters are updated for a single epoch. The same model is updated for each *goID* until the iterations through all *goID*s are complete and we obtain the pre-trained model.

2) Phase 2: Fine-Tuning: Moving ahead to the fine-tuning phase, as shown in Fig. 3b, we follow a similar process. However, this time instead of initializing a new model, we use the pre-trained model from phase 1. Further, rather than updating the model weights, we update the weights and save as a separate model for each *goID* resulting in N fine-tuned models-one per *goID*. We perform the training for a maximum of 100 epochs with an early stopping based on validation dataset. These fine-tuned models are the final version of the model used for input reconstruction.

3) Detection: While Fig 2a shows how the batch processing of inputs across all *goIDs,* Figure 4 demonstrates detection pipeline for a single GOOSE packet. First, features are extracted from the new packet, similar to the process shown in Fig 2a. The process uses the packet's unique *goID* to retrieve the corresponding fine-tuned model and previous temporal data. The new packet features are appended to the previous temporal data to form the input temporal window. This window is then passed through a SoftMax layer, forward passed through finetuned reconstruction model and mean square error is calculated to get the reconstruction error. Finally, this error is then compared against previously defined thresholds obtained through DiFAD part of the model. If the reconstruction error exceeds this threshold, the packet is flagged as an anomaly.

TABLE II
PR AUC AND ROC AUC OF MODELS FOR POWERDUCK AND SYNTHESIZED DATASET WITH APPROXIMATE TESTING TIME

	Powerduck Dataset[30]		Synthesized dataset[31]		Approx. forward pass
Model	ROC AUC	PR AUC	ROC AUC	PR AUC	latency per frame(window)
CNN [12]	.9569 ± .006	.7546 ± .010	.9386 ± .007	.4763 ± .001	2.5329ms
Ensemble [13]	.9091 ± .020	.6328 ± .044	.9541 ± .005	.4930 ± .004	2.8226ms
LSTM [14]	.9543 ± .001	.7517 ± .001	.9545 ± .000	.4965 ± .000	**2.3899ms**
Transformer	.9560 ± .003	.7524 ± .013	.9542 ± .004	**.4973 ± .000**	3.4409ms
ResNeST [15]	.5144 ± .115	.5122 ± .102	.5568 ± .094	.0108 ± .028	3.5213ms
TF-DiFAD	**.9684 ± .002**	**.7666 ± .013**	**.9573 ± .003**	**.4973 ± .000**	3.4409ms

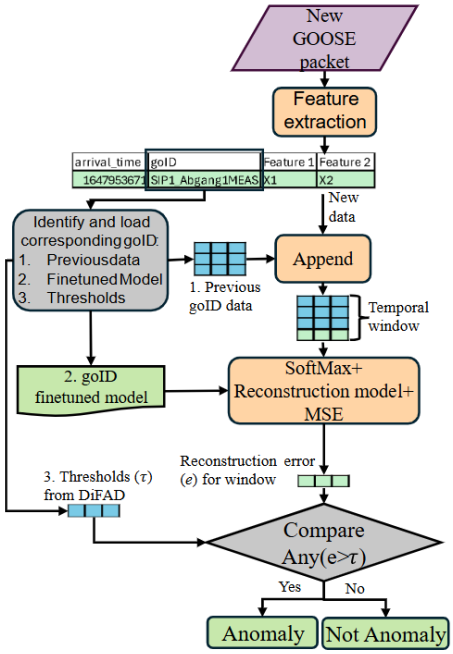


Fig. 4. Fine-tuned model and thresholds used for detecting anomaly in a GOOSE packet.

IV. ANOMALY DETECTION PERFORMANCE EVALUATION

A. Test Dataset Description

The model is evaluated using two substation datasets. The first is the synthesized dataset developed by Biswas et al. [29]. The second dataset is the PowerDuck dataset published by Zemanek et al. [30]. Both datasets are publicly available. No actual substation infrastructure data is being used.

The synthesized dataset [29] simulates a 66/11kV substation using the IEC 61850 protocols. The network comprises of 18 Intelligent Electronic Devices (IEDs) including 14 line IEDs, 2 transformer IEDs, 1 bus IED, and 1 under frequency IED. Each IED generates three streams of data which are, status, which is responsible for control; alarm, which ensures protection, and measurement, which provides physical layer measurements. The dataset includes a variety of cyber attacks designed to manipulate GOOSE data frames. It consists of two types of suppression attacks, three distinct data manipulation attacks, and a composite attack that combines multiple attacks and two types of flooding attacks.

The Powerduck dataset [30] is comparatively smaller in terms of the number of nodes, but the data is collected from real IEDs as opposed to simulated IEDs in synthesized dataset. The Powerduck dataset also considers multiple cyber attack scenarios. These include three types of replay attacks, five types of injection or data manipulation attacks, five types of suppression attacks, and a network flooding attack. In our work, we do not consider the events and flood types of data, keeping them out of scope.

B. Model Training and Testing

We compare the proposed TF-DiFAD against several state-of-the-art anomaly detection models that use reconstruction techniques for power systems as presented in Table II, Table III and Table IV, i.e., CNN [12], Ensemble [13], LSTM [14] and ResNeST [15]. Due to the unavailability of original source code from previous publications, each model is reimplemented and adapted to our dataset, with hyperparameter tuning performed for most of the models to achieve optimal results. The performance of these models is calculated assuming the conventional thresholds of normally distributed reconstruction error.

Firstly, we implement the CNN-autoencoder approach from [12], originally used for cyber attack detection in protective relays. This implementation follows the publication methodology and doesn't require additional hyperparameter tuning. Second, we implement the RNN+GRU+LSTM ensemble model from [13] for security monitoring of GOOSE frames. Since the original model was used for prediction as opposed to reconstruction, we perform hyperparameter tuning to maximize the model's performance. Third, we compare it with our previously proposed LSTM model [14], designed for reconstructing OT network traffic. Since our approach is unsupervised and only uses digital substation data, the CNN and graph convolution parts are excluded. Hyperparameter optimization is performed to achieve optimal performance. Fourth, we implement a vanilla transformer model without distribution fitting to evaluate the impact of adding the DiFAD layer in our proposed TF-DiFAD model. Lastly, we implement the ResNeSt model introduced in [15] previously applied to substation-level traffic. Unlike other models, the ResNeSt model uses single-point input rather than windowed sequences. Additionally, we do not implement the FARIMA-based extension from the original due to its incompatibility with our architecture. Except for ResNeSt, all models are trained using windowed input sequences alongside the SoftMax layer as described previously. Also, we use the two-phase training strategy for all the models to have a consistent and fair comparison.

We train our model using the Adam optimizer which adjusts the learning rates during training. We start with an initial

TABLE III
PERFORMANCE COMPARISON OF MODELS FROM LITERATURE WITH PROPOSED TF-DIFAD MODEL FOR POWERDUCK DATASET

Threshold	Model	Balanced Acc.	Youden's J	DOR	MCC	F1 Score
CDF≈ .99865 (μ+3σ equivalent of normal distribution)	CNN [12]	.9359 ± .003	.8718 ± .007	219.4 ± 21.2	.6669 ± .018	.6666 ± .019
	Ensemble [13]	.8648 ± .030	.7296 ± .060	74.63 ± 38.3	.4418 ± .075	.4218 ± .080
	LSTM [14]	.9297 ± .001	.8595 ± .003	195.1 ± 12.5	.6245 ± .001	.6185 ± .002
	Transformer	.9378 ± .003	.8756 ± .005	288.3 ± 67.5	.6496 ± .039	.6443 ± .046
	ResNeST [15]	.5463 ± .106	.0927 ± .212	.5019 ± 1.55	.0710 ± .120	.1535 ± .034
	TF-DiFAD	**.9554 ± .005**	**.9109 ± .010**	**1373 ± 83.3**	**.6845 ± .029**	**.6772 ± .033**
CDF≈ .99996 (μ+4σ equivalent of normal distribution)	CNN [12]	.9216 ± .003	.8432 ± .007	152.8 ± 16.8	.6891 ± .016	.6988 ± .017
	Ensemble [13]	.8608 ± .025	.7216 ± .051	48.72 ± 30.4	.4607 ± .079	.4520 ± .085
	LSTM [14]	.9246 ± .001	.8492 ± .002	164.4 ± 7.69	**.6967 ± .009**	**.7062 ± .009**
	Transformer	.9305 ± .009	.8611 ± .019	201 ± 66.0	.6746 ± .022	.6780 ± .025
	ResNeST [15]	.5463 ± .106	.0930 ± .212	.5019 ± 1.55	.0710 ± .120	.1535 ± .034
	TF-DiFAD	**.9339 ± .022**	**.8678 ± .044**	**363.6 ± 293**	.6716 ± .019	.6729 ± .012

TABLE IV
PERFORMANCE COMPARISON OF MODELS FROM LITERATURE WITH PROPOSED TF-DIFAD MODEL FOR SYNTHESIZED SECURITY DATASET

Threshold	Model	Balanced Acc.	Youden's J	DOR	MCC	F1 Score
CDF≈.99865 ($\mu+3\sigma$ equivalent of normal distribution)	CNN [12]	$.9616 \pm 3.7e-3$	$.9231 \pm 7.3e-3$	$732. \pm 196$	$.0499 \pm 6.6e-3$	$.0055 \pm 1.4e-3$
	Ensemble [13]	$.9711 \pm 2.9e-4$	$.9422 \pm 5.8e-4$	2426 ± 185	$.0913 \pm 3.4e-3$	$.0176 \pm 1.0e-3$
	LSTM [14]	$.9715 \pm 4.0e-6$	$.9429 \pm 8.0e-6$	2661 ± 3.0	$.0957 \pm 5.3e-5$	$.0192 \pm 2.1e-5$
	Transformer	$.9704 \pm 1.1e-4$	$.9408 \pm 2.2e-4$	2058 ± 51	$.0842 \pm 1.0e-3$	$.0150 \pm 3.6e-4$
	ResNeST [15]	$.5000 \pm .0$	$.0 \pm .0$	$.0 \pm .0$	$.0 \pm .0$	$.0 \pm .0$
	TF-DiFAD	**$.9730 \pm 5.3e-5$**	**$.9460 \pm 1.1e-4$**	**4769 ± 129**	**$.1276 \pm 1.7e-3$**	**$.0338 \pm 8.8e-4$**
CDF≈ .99996 ($\mu+3\sigma$ equivalent of normal distribution)	CNN [12]	$.9679 \pm 1.6e-3$	$.9358 \pm 3.3e-3$	$1389. \pm 321$	$.0689 \pm 8.0e-3$	$.0102 \pm 2.3e-3$
	Ensemble [13]	$.9733 \pm 1.6e-4$	$.9465 \pm 3.1e-4$	$5459. \pm 483$	$.1363 \pm 5.9e-3$	$.0385 \pm 3.3e-3$
	LSTM [14]	$.9734 \pm 3.1e-6$	$.9468 \pm 6.2e-6$	5828 ± 11.1	$.1408 \pm 1.3e-4$	$.0410 \pm 7.5e-5$
	Transformer	$.9734 \pm 4.6e-5$	$.9469 \pm 9.2e-5$	6064 ± 179	$.1436 \pm 2.1e-3$	$.0426 \pm 1.2e-3$
	ResNeST [15]	$.5000 \pm .0$	$.0 \pm .0$	$.0 \pm .0$	$.0 \pm .0$	$.0 \pm .0$
	TF-DiFAD	**$.9742 \pm 3.0e-5$**	**$.9483 \pm 6.0e-5$**	**11350 ± 406**	**$.1946 \pm 3.3e-3$**	**$.0767 \pm 2.5e-3$**

learning rate of 0.001 combined with a dynamic learning rate scheduler called *ReduceOnPlateau*. The learning rate is adjusted based on model convergence and number of epochs. In case of a larger learning rate, the loss would increase to arbitrary large values due to the overshoot in gradient making the loss divergent, while a smaller learning rate would take more epochs for the model to converge. The window size is set to 30 and the batch size is 400, with batch shuffling enabled. Early stopping is implemented if the change in validation loss (MSE loss) falls below 10^{-6} and if absolute validation loss goes below 0.0002. The hyperparameters of the transformer model chosen for optimal performance are as follows:

- Embedding layer dimensions: 256
- Number of attention heads: 32
- Number of encoder and decoder layers: 4
- Dropout rate: 0.2

With the above hyperparameters, the overall trainable parameter count for the transformer adds up to 11,582,219. We use approximately 50% of the normal data for training the model, 20% for validation and the remaining 30% is reserved for testing. The testing dataset includes 30% of normal data as mentioned previously and all of the attack data, both of which are never used during any step of the training process.

C. Evaluation Metrics

The models are evaluated using multiple metrics. However, due to the highly imbalanced nature of our dataset, with less than 5% attack data, we place less emphasis on general metrics such as accuracy, precision, recall (also known as sensitivity), specificity, and F1 score. Instead, we focus on evaluation metrics, better suited to highly imbalanced datasets. Based on the confusion matrix, which includes True Positives (TP), False Positives (FP), True Negatives (TN) and False Negatives (FN) the selected metrics are defined as follows:

1) *ROC curve*: Receiver Operating Characteristics curves show the classification results of the models based on different thresholds. The area under this curve is known as ROC-Area Under Curve (*ROC AUC*). Higher ROC AUC implies better class distinguishability.
2) *PR-Curve*: Precision Recall curve plots the precision and recall of the model at various thresholds to give the effect of tradeoff between precision and recall with moving threshold. The area under this curve is known as PR-Area Under Curve (*PR AUC*). Higher PR AUC implies better precision and recall at different thresholds.
3) *Balanced Accuracy*: It is the arithmetic mean of specificity and sensitivity, and it helps find the accuracy of the model in an imbalanced dataset.

$$Balanced\ Acc. = \frac{Sensitivity + Specificity}{2}$$

4) Youden's J: Youden's J Index (J) Helps in Finding an Optimal Point in the ROC Curve [31]

$$J = Sensitivity + Specificity - 1$$

5) *Diagnostic Odds Ratio (DOR)*: DOR is generally used in medical fields and it calculates the ratio of the odds

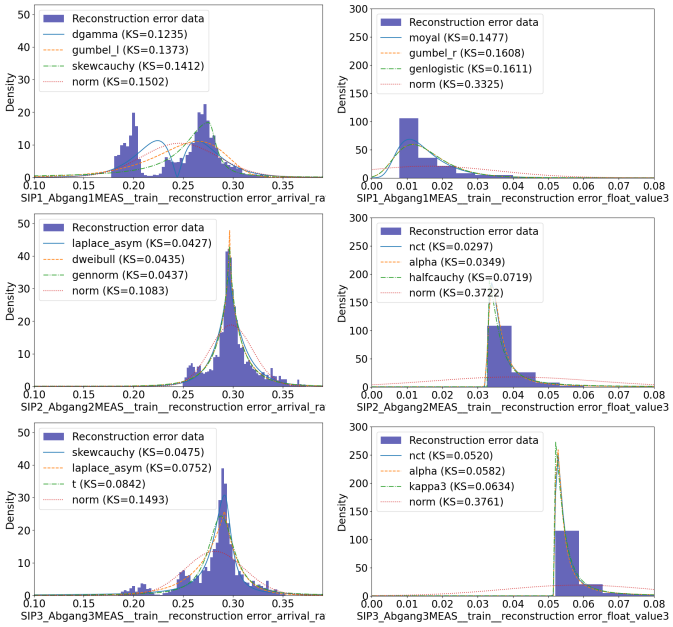


Fig. 5. Reconstruction error distribution and corresponding distribution curves fitted for two features for three *goID*s listed with increasing K-S score with Normal distribution having the highest (worst) K-S score.

of positivity in disease (anomaly for us) to the odds of positivity in the non-diseased (normal for us) [32].

$$DOR = \frac{TP}{FN} / \frac{FP}{TN}$$

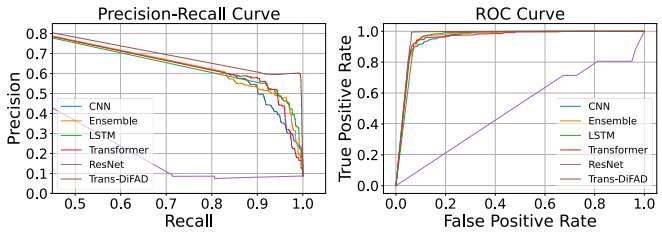
6) *Matthews Correlation Coefficient (MCC)*: MCC is useful for unbalanced datasets as it produces high value only if the model can predict majority of positive data and majority of negative data correctly.

$$MCC = \frac{(TP.TN) - (FP.FN)}{\sqrt{(TP+FP).(TP+FN).(TN+FP).(TN+FN)}}$$

D. Results

Once the hyperparameters of the models are tuned for optimal results, we train each model from scratch at least 10 times, covering both pretraining and fine-tuning phases. We then compute evaluation metrics for each of these 10 runs, and provide their mean and standard deviation. This provides a more reliable performance measure, considering model's random initialization. The entire process is performed on the Delft Blue supercomputer [33] using one GPU (10GB video RAM) and two CPU cores (5333MB RAM each). Moreover, to test the deploy-ability of the model in field, we run the TF-DiFAD model on a windows workstation with 4 cores, 64GB RAM and 4GB dedicated video RAM.

As mentioned previously, we calculate the reconstruction error between the input and reconstructed output from the transformer model using MSE. Figure 5 shows the distributions of reconstruction errors for two of the 11 features (arrival rate and float payload) extracted from three *goID*s from the Powerduck dataset. We fit 34 candidate distributions to these error distributions using MLE and compare their K-S statistics. In the unlikely case of the reconstruction errors being so anomalous that it cannot be fitted to any of these 34 distributions, the Gaussian Normal distribution is still one of the candidates. As most of the data can be fitted to a Gaussian curve, the model remains stable and would produce anomaly scores. Figure 5 demonstrates how well some of the candidate distributions fit these error distributions and their corresponding K-S statistics. We observe that the common assumption of reconstruction errors following a normal distribution is invalid, since the normal distribution exhibits the highest K-S statistic for both features, indicating it provides the poorest fit. In contrast, other distributions analyzed in this study model the reconstruction errors more accurately.



(a) PR Curve for considered models (Zoomed). (b) ROC Curve for considered models.

Fig. 6. Comparison of model performance using Precision-Recall and ROC curves for Powerduck dataset.

Next, we analyze the model's final outputs. Although each data point in the dataset is labeled as anomalous or normal, we classify an entire sliding window of 30 data points as anomalous if any of those data points are labeled as anomalous by the model. Because a single anomalous data point remains within the sliding window for 30 consecutive steps, each window containing it is marked anomalous. As a result, the next 30 windows end up being labeled anomalous. Although this could pose challenges in practical applications, localizing anomalies in the window is beyond the scope of this paper.

We analyze one of the model outputs, which is the probability scores for each data point and use them to plot PR and ROC curves for all the models, as shown in Fig 6. A PR curve closer to the upper right corner and an ROC curve closer to the upper left corner indicate better model performance. Among the models evaluated, the proposed TF-DiFAD model produces the best PR and ROC curves. In contrast, the ResNeST model, which uses single-point data rather than windowed data, exhibits the weakest performance in all aspects. A further reason for the bad performance of the model could be the lack of implementation of the FARIMA part of the algorithm, which was incompatible with our model. This is also reflected in PR AUC and ROC AUC as shown in Table II, the proposed model provides the highest ROC AUC of 0.9684 and 0.9573 for the Powerduck and synthesized dataset, respectively. This implies that the model has a strong capacity to distinguish anomalies and normal data. Further, the PR AUC of the proposed model is 0.7666 and 0.4973, which implies that there are still some data that cannot be perfectly classified as anomalous or normal. This is due to the aforementioned issue with the windowed approach. However, the proposed TF-DiFAD model performs better than all other models in PR and ROC aspects.

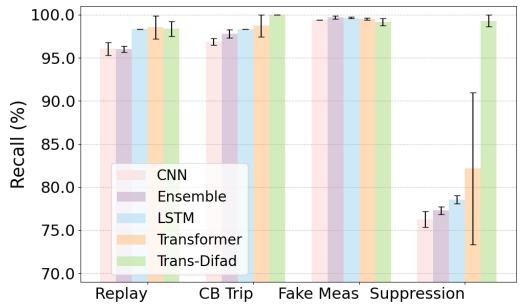


Fig. 7. Powerduck dataset recall for individual attacks.

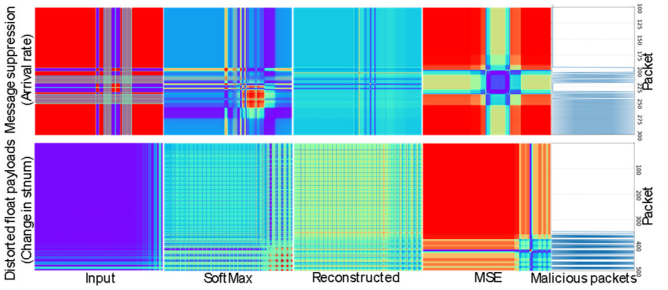


Fig. 8. GAF heatmaps to visualize input data changes after the applied layers.

Table II also shows the approximate forward pass latency per window, calculated on Delft Blue supercomputer which is also used for training. The LSTM model achieves the best latency of 2.3899 milliseconds per window due to its simple design. The proposed model, the TF-DiFAD model, gives a latency of 3.441 milliseconds, which is not significantly higher than others.

Moving ahead, we examine the second output from the model, which is the binary prediction 0/1, implying normal or anomaly. We generate a confusion matrix from these predictions and calculate the aforementioned metrics. We compare the model's performance under two thresholds with CDFs equal to $\mu + 3\sigma$ and $\mu + 4\sigma$ of normal distribution. Although both thresholds are considered, $\mu + 3\sigma$ is the most used default threshold for anomaly detection when no prior information about the anomaly is available. Table III shows the performance of the selected models for the Powerduck dataset. The proposed model outperforms all other models in $\mu+3\sigma$ equivalent threshold for all the metrics considered. For $\mu + 4\sigma$ equivalent threshold, it is better on three out of five metrics.

Table IV shows the performance of the selected models for the synthesized dataset. For this dataset, the proposed model again outperforms all other models on every metric under both thresholds. We also see that the $\mu + 4\sigma$ threshold performs slightly better than $\mu+3\sigma$. It is important to note that this dataset has an extremely high imbalance in the classes, which is reflected in the low MCC and F1-score across all the models. Finally, we calculate the recall for individual attacks in the Powerduck dataset to assess how effectively each model detects different attack types. We exclude the ResNeST model due to its subpar performance. As shown in Fig. 7, most models successfully detect replay, circuit breaker trip, and fake measurement injection attacks but are failing to identify suppression attacks. A suppression attack occurs when the attacker injects fake measurements with a high-status number but does not modify the payload, causing the IED to discard legitimate new packets. The better performance of the model against suppression attacks shows that the model is evaluating not only the payload but also the patterns in the rate of arrival, status number, and sequence number to detect anomalies.

For testing the local deployment, we assess the resources consumption of the model on the workstation. We find the approximate forward pass latency to be 3.4927ms per frame. Translating to a throughput of 286FPS. In terms of memory overhead, the process utilized peaked at 2571 MB (increasing from an idle baseline of 13,719.6MB to 16,291MB). Similarly, the dedicated video RAM usage peaked at 3671MB, rising from a baseline of 191MB to 3862MB during the process. This proves the model's deploy-ability in local environment without heavy computation overhead.

E. Model Interpretability

Finally, to enhance model interpretability and understand how different layers affect the input data, Gramian Angular Field (GAF) heatmaps are used to visualize the data after transformations. The visualization include the input frame, consisting of extracted features, the input after SoftMax, the reconstructed data from the model, and the reconstruction error (MSE). Additionally, the instance at which malicious packets were injected in the substation network is indicated in the final plot. Fig. 8 shows the GAF heatmaps for the message suppression attack (top row) with the arrival rate feature, and the distorted float payload (bottom row) with the change in *stnum* feature. It can be observed how the data is transformed as it passes through each layer. While it is clear from the extracted feature that a message suppression attack has begun, the beginning of the distorted payload attack is not immediately obvious. However, after the SoftMax function, it becomes clearer as the changes in the data are exaggerated. The Reconstructed feature shows how the model should behave considering a normal operating case, while the MSE represents the error between the SoftMax and the reconstructed feature. We see that the onset of the attack is quite apparent for both cases in the MSE visualization. However, for the message suppression attack, the arrival rate feature is unable to consistently identify malicious packets after a certain time, and is identified by other features.

V. Conclusion and Future Work

The paper introduces TF-DiFAD, an unsupervised transformer and distribution fitting-based anomaly detection method using GOOSE deep packet inspection in digital substations. It uses the state-of-the-art transformer model to detect anomalies and defines a technique for setting outlier thresholds and calculating the anomaly score while considering the GOOSE data frame information from multiple layers and payload. The proposed distribution fitting algorithm accurately models the reconstruction error distribution reflected by a better K-S statistic as compared to the common assumption of normal distribution. The classification method successfully identifies the anomalies with high accuracy even when the anomaly data is extremely rare and outperforms state-of-the-art models including CNN, LSTM and Ensemble approaches on metrics such as ROC AUC and PR AUC. Notably, it achieves

a 98-99% recall, ensuring that majority of cyber attacks are being detected along with ROC AUC of 95-96%, ensuring that model performs good at multiple thresholds. The model keeps the forward pass latency low at 3.44-3.5ms per window while performing best at most of the unbalanced performance indicator metrices such as balanced accuracy, Youden's J, MCC and DOR.

While, the current study focuses on anomalies only in GOOSE messages, future work can be extended to Sampled Values by under sampling and parsing as well as complex payloads of MMS. The current offline or forensic work could also be extended for real-time anomaly detection in continuous data streams and be tested in an emulated hardware-in-the-loop setup. Additionally, the model's applicability would be enhanced by pinpointing the exact data points rather than the window. To address the independence assumption, we can model the joint probability and correlations between reconstruction error features. Finally, although the forward pass latency of the proposed model is comparable to other models, the training time is significantly higher and needs further research to optimize.

REFERENCES

[1] R. Hunt, B. Flynn, and T. Smith, "The substation of the future: Moving toward a digital solution," *IEEE Power Energy Mag.*, vol. 17, no. 4, pp. 47–55, Jul. 2019.

[2] J. Gaspar, T. Cruz, C.-T. Lam, and P. Simões, "Smart substation communications and cybersecurity: A comprehensive survey," *IEEE Commun. Surv. Tut.*, vol. 25, no. 4, pp. 2456–2493, 4th Quart., 2023.

[3] H. M. Khalid et al., "WAMS operations in power grids: A track fusion-based mixture density estimation-driven grid resilient approach toward cyberattacks," *IEEE Syst. J.*, vol. 17, no. 3, pp. 3950–3961, Sep. 2023.

[4] H. M. Khalid, F. Flitti, M. S. Mahmoud, M. M. Hamdan, S. M. Muyeen, and Z. Y. Dong, "Wide area monitoring system operations in modern power grids: A median regression function-based state estimation approach towards cyber attacks," *Sustain. Energy, Grids Netw.*, vol. 34, Jun. 2023, Art. no. 101009.

[5] M. Liu, C. Zhao, Z. Zhang, R. Deng, P. Cheng, and J. Chen, "Converter-based moving target defense against deception attacks in DC microgrids," *IEEE Trans. Smart Grid*, vol. 13, no. 5, pp. 3984–3996, Sep. 2022.

[6] S. Peng, M. Liu, L. Chai, and R. Deng, "DST-GNN: A dynamic spatiotemporal graph neural network for cyberattack detection in grid-tied photovoltaic systems," *IEEE Trans. Smart Grid*, vol. 16, no. 1, pp. 330–343, Jan. 2025.

[7] A. A. Nassar and W. G. Morsi, "Detection of cyber-attacks and power disturbances in smart digital substations using continuous wavelet transform and convolution neural networks," *Electric Power Syst. Res.*, vol. 229, Apr. 2024, Art. no. 110157.

[8] V. E. Quincozes, S. E. Quincozes, D. Passos, C. Albuquerque, and D. Mossé, "Towards feature engineering for intrusion detection in IEC–61850 communication networks," *Ann. Telecommun.*, vol. 79, nos. 7–8, pp. 537–551, Feb. 2024.

[9] X. Wang, C. Fidge, G. Nourbakhsh, E. Foo, Z. Jadidi, and C. Li, "Anomaly detection for insider attacks from untrusted intelligent electronic devices in substation automation systems," *IEEE Access*, vol. 10, pp. 6629–6649, 2022.

[10] J. Liu et al., "Deep anomaly detection in packet payload," *Neurocomputing*, vol. 485, pp. 205–218, May 2022.

[11] *Power Systems Management and Associated Information Exchange—Data and Communications Security—Part 6: Security*, Standard 62351-6:2020, 2020.

[12] Y. M. Khaw, A. Abiri Jahromi, M. F. M. Arani, S. Sanner, D. Kundur, and M. Kassouf, "A deep learning-based cyberattack detection system for transmission protective relays," *IEEE Trans. Smart Grid*, vol. 12, no. 3, pp. 2554–2565, May 2021.

[13] A. Albarakati et al., "Security monitoring of IEC 61850 substations using IEC 62351–7 network and system management," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 1641–1653, Mar. 2022.

[14] A. Presekal, A. Ştefanov, V. S. Rajkumar, and P. Palensky, "Attack graph model for cyber-physical power systems using hybrid deep learning," *IEEE Trans. Smart Grid*, vol. 14, no. 5, pp. 4007–4020, Sep. 2023.

[15] T. Yang, Y. Hou, Y. Liu, F. Zhai, and R. Niu, "WPD-ResNeSt: Substation station level network anomaly traffic detection based on deep transfer learning," *CSEE J. Power Energy Syst.*, pp. 1–12, Nov. 2021.

[16] O. Koren, M. Koren, and O. Peretz, "A procedure for anomaly detection and analysis," *Eng. Appl. Artif. Intell.*, vol. 117, Jan. 2023, Art. no. 105503.

[17] Z. Zamanzadeh Darban, G. I. Webb, S. Pan, C. Aggarwal, and M. Salehi, "Deep learning for time series anomaly detection: A survey," *ACM Comput. Surv.*, vol. 57, no. 1, pp. 1–42, Jan. 2025.

[18] J. Suaboot et al., "A taxonomy of supervised learning for IDSs in SCADA environments," *ACM Comput. Surv.*, vol. 53, no. 2, pp. 1–37, Mar. 2021.

[19] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surv. Tut.*, vol. 20, no. 4, pp. 3369–3388, 4th Quart., 2018.

[20] D. Jay, H. Goyel, U. Manickam, and G. Khare, "Unsupervised learning based intrusion detection for GOOSE messages in digital substation," in *Proc. 22nd Nat. Power Syst. Conf. (NPSC)*, Dec. 2022, pp. 242–247.

[21] I. Ji, S. Jeon, and J. T. Seo, "AE-LSTM based anomaly detection system for communication over DNP 3.0," in *Information Security Applications (WISA 2023) (Lecture Notes in Computer Science)*. Singapore: Springer, 2024, pp. 91–104, doi: 10.1007/978-981-99-8024-6_8.

[22] W. Hao, Q. Yang, Z. Li, S. Hu, B. Liu, and W. Ruan, "Multi-scale traffic aware cybersecurity situational awareness online model for intelligent power substation communication network," *IEEE Internet Things J.*, vol. 10, no. 2, pp. 1666–1681, Jan. 2023.

[23] A. Presekal, A. Ştefanov, I. Semertzis, and P. Palensky, "Spatio-temporal advanced persistent threat detection and correlation for cyber-physical power systems using enhanced GC-LSTM," *IEEE Trans. Smart Grid*, vol. 16, no. 2, pp. 1654–1666, Mar. 2025.

[24] S. R. Firouzi, L. Vanfretti, A. Ruiz-Alvarez, H. Hooshyar, and F. Mahmood, "Interpreting and implementing IEC 61850–90–5 routed-sampled value and routed-GOOSE protocols for IEEE C37.118.2 compliant wide-area synchrophasor data transfer," *Electr. Power Syst. Res.*, vol. 144, pp. 255–267, Mar. 2017.

[25] A. Vaswani et al., "Attention is all you need," 2017, *arXiv:1706.03762*.

[26] W. H. Press, *Numerical Recipes: The Art of Scientific Computing*. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[27] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. (2010). *Why Does Unsupervised Pre-Training Help Deep Learning?*. [Online]. Available: http://jmlr.org/papers/v11/erhan10a.html

[28] J. Nguyen, J. Wang, K. Malik, M. Sanjabi, and M. Rabbat, "Where to begin? On the impact of pre-training and initialization in federated learning," 2022, *arXiv:2210.08090*.

[29] P. P. Biswas, H. C. Tan, Q. Zhu, Y. Li, D. Mashima, and B. Chen, "A synthesized dataset for cybersecurity study of IEC 61850 based substation," in *Proc. IEEE Int. Conf. Commun., Control, Comput. Technol. Smart Grids (SmartGridComm)*, Oct. 2019, pp. 1–7.

[30] S. Zemanek, I. Hacker, K. Wolsing, E. Wagner, M. Henze, and M. Serror, "PowerDuck: A GOOSE data set of cyberattacks in substations," in *Proc. 15th Workshop Cyber Secur. Experimentation Test*, Virtual, CA, USA, 2022, pp. 49–53.

[31] G. Siva, "Matrix variate receiver operating characteristic curve for binary classification," *Statistics*, vol. 58, no. 1, pp. 1–8, Jan. 2024.

[32] A. S. Glas, J. G. Lijmer, M. H. Prins, G. J. Bonsel, and P. M. M. Bossuyt, "The diagnostic odds ratio: A single indicator of test performance," *J. Clin. Epidemiology*, vol. 56, no. 11, pp. 1129–1135, Nov. 2003.

[33] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. Accessed: Jan. 20, 2025. [Online]. Available: https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2